UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

19|20

Trabajo fin de grado

Bosques contextuales: Una nueva técnica para desambiguación lingüística

Daniel Guzmán Olivares

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO FIN DE GRADO

Bosques contextuales: Una nueva técnica para desambiguación lingüística

Autor: Daniel Guzmán Olivares
Tutor: Lara Quijano Sánchez

julio 2020

Daniel Guzmán Olivares
Bosques contextuales: Una nueva técnica para desambiguación lingüística

Daniel Guzmán Olivares
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

A Gloria, por haber estado siempre a mi lado

Bilbo Bolsón - ¡Buenos días!
Gandalf - ¿Qué quieres decir? ¿Me deseas un buen día, o quieres decir que es un buen día, lo quiera yo o no? ¿O quieres decir que te sientes bien esta mañana en particular? ¿O simplemente afirmas que es una mañana en la que hay que sentirse bien?

J.R.R. Tolkien

Agradecimientos

En primer lugar me gustaría expresar mi agradecimiento a toda mi familia y amigos, con una mención especial a Gloria del Valle, por el apoyo y la comprensión durante la realización de este trabajo, ya que además de haber estado a mi lado durante los últimos cinco años de carrera, estos últimos meses han aguantado conversaciones sobre "bosques"que se expanden y desambiguación de palabras en mayor medida de lo que me gustaría admitir por escrito.

Me gustaría también destacar el trabajo de mi tutora, la Dra. Lara Quijano Sánchez por todos los recursos y ayuda que me ha prestado durante este curso para poder sacar adelante mis ideas.

Finalmente me gustaría agradecer a todos los profesores que he tenido durante mi trayectoria académica en la universidad, pues si he podido construir un modelo como este ha sido gracias a la formación que he recibido estos años.

Resumen

En el Procesamiento del Lenguaje Natural, el problema de la desambiguación lingüística consiste en encontrar la acepción adecuada de una palabra polisémica dentro del contexto de una frase. La investigación de este problema ha puesto en evidencia que los mejores resultados para este problema están estrechamente relacionados con la utilización de conocimiento externo o modelos preentrenados como las últimas variaciones de BERT y GPT-2. Este trabajo tiene como objetivo introducir los bosques contextuales, una nueva técnica basada en modelos gráficos probabilísticos usando conocimiento externo en tiempo real para demostrar que potencialmente puede mejorar el entendimiento de las palabras polisémicas.

Para responder a esta pregunta se van a realizar dos experimentos comparando el rendimiento de los bosques contextuales contra BERT y GloVe sobre un dataset con 75 frases cortas en inglés con palabras polisémicas. Los resultados de estos experimentos mostrarán la superioridad de los árboles contextuales de la que se seguirá que los modelos de minado de información en tiempo real usando un enfoque basado en modelos gráficos probabilísticos son una aproximación válida para el problema de la desambiguación lingüística.

Palabras clave

desambiguación lingüística, bosques contextuales, contexto, modelos gráficos, conocimiento externo, modelos probabilísticos, procesamiento de lenguaje natural, NLP

Abstract

In Natural Language Processing, word sense disambiguation problem aims to find the correct interpretation of a polysemic word in a phrase. Research has shown that the best results for this problem are intimately related to the use of external knowledge or pre-trained models such as the latest variations of BERT or GPT-2. This study aims to present Contextual Forests, a new technique based on probabilistic graphical models and real-time knowledge retrieving in order to prove that it can potentially make an improvement on polysemic word understanding.

In order to prove these results, two experiments will be conducted to test contextual forest performance against BERT and GloVe using a dataset composed of 75 short phrases with polysemic words in all of them. The results of this experiment provide evidence of the superiority of the contextual forests leading to the conclusion that real-time information retrieving models in a probabilistic graphical approach are a valid approach for the word sense disambiguation problem.

Keywords

word sense disambiguation, WSD, contextual forests, context, graphical models, external knowledge, probabilistic models, Natural Language Processing, NLP

Índice

Listas

Lista de algoritmos

Lista de códigos

Lista de cuadros

Lista de ecuaciones

Lista de figuras

Lista de tablas

Lista de cuadros

1

Introducción y Estado del Arte

1.1. Introducción

Vivimos en la era de la información, la cantidad de datos a nuestra disposición ha crecido exponencialmente con los años así como el interés en su estudio, interés que se ha visto reflejado en el auge de los campos dedicados al procesamiento y análisis de estos datos.
Este trabajo está enfocado en uno de esos campos, el *Procesamiento de Lenguaje Natural (NLP)*, que tiene como objetivo el estudio de las interacciones de una computadora con el lenguaje humano. En particular dentro de este amplio campo, se estudiará el problema de la desambiguación lingüística.
El problema de la desambiguación lingüística consiste en encontrar la acepción adecuada de una palabra polisémica en una frase usando su contexto y se ilustra fácilmente con un ejemplo; consideremos la siguiente frase:

"Tengo que ir al banco a sacar dinero "

En esta frase, la palabra polisémica *banco* se refiere al emplazamiento de una institución financiera, algo que cualquier humano deduce con facilidad. Pero, ¿y una computadora? ¿Sería capaz de reconocer que *banco* se refiere a un emplazamiento o tomaría cualquier otra acepción? ¿Podemos construir un algoritmo que cumpla con este propósito?
La respuesta a esta última pregunta no es ni mucho menos trivial. A lo largo de los años se han propuesto diferentes modelos para resolver este problema, empezando por los primeros en 1980 basados en reglas morfosintácticas (Fass and Wilks [1983]) y pasando por los modelos basados en *Machine Learning* de finales de los 90 (Yarowsky [1995]), hasta finalmente llegar a los modelos de hoy en día.
El interés de los investigadores en la desambiguación lingüistica viene motivado porque este problema en particular forma parte de tareas como la traducción automatizada, la extracción de información, el estudio sintáctico automatizado, la lexicografía y la extracción de palabras clave (Kilgarriff [1997]).
La mayoría de los modelos que están ofreciendo los mejores resultados a fecha de hoy, como *BERT* (Devlin et al. [2019]) y *GPT-2* (Radford et al. [2019]) están basados en un aprendizaje no supervisado o semi-supervisado para generar representaciones vectoriales (*word embeddings*) dependientes del contexto.

En este trabajo se propone un modelo de desambiguación lingüística independiente del idioma para frases cortas. Este modelo no está basado en redes neuronales para generar *word embeddings* sino en métodos estadísticos y probabilísticos apoyados por búsqueda dinámica de conocimiento externo en Wikipedia. Esto supondrá un avance en el sentido de no tener que pre-entrenar los modelos cada vez que cambien las relaciones entre términos del corpus (imaginemos la relación existente entre las palabras *Coronavirus* y *China* hace 10 meses y a día de hoy) y poder conservar la propiedad de tomar los conceptos "más cercanos" a uno dado, que *BERT* y *GPT-2* pierden a la hora de hacer sus *word embeddings* dependientes de contexto (esto se aclarar á más adelante después de profundizar en el funcionamiento interno de BERT en el capítulo 4).

El resto de la memoria está estructurada de la siguiente forma:
El modelo se explicará con detalle en el capítulo 2, previamente se revisará el estado del arte en el problema de la desambiguación lingüística en la siguiente sección.
En el capítulo 3 se expondrá el modelo de los arboles contextuales.
En el capítulo 4 se explicarán los experimentos diseñados para comparar el rendimiento del modelo contra otras técnicas y sus resultados.
Por último, el capítulo 5 presenta las conclusiones finales y el trabajo para realizar en el futuro.
Todo el código utilizado tanto para el desarrollo como para los experimentos se puede encontrar en el Github DanielGuzmanOlivares/ContextualForests

2

Estado del Arte

Como se expuso en la sección anterior, los primeros modelos de desambiguación estaban basados en reglas de aprendizaje y evolucionaron a métodos basados en Aprendizaje Automático supervisado en los años 90. A principios de los 2000 los modelos basados en aprendizaje automático alcanzaron una cierta estabilidad en rendimiento y se empezaron a utilizar métodos basados en aprendizaje semi-supervisado y no supervisado (Edmonds and Agirre [2008]).
Aunque fue a partir de la aparición de *word2vec* (Mikolov et al. [2013]) a finales de 2013 cuando los métodos basados en *word embedding* empezaron a ganar popularidad (ver Figura 2.2) y se pueden distinguir dos corrientes que se siguen manteniendo a día de hoy:

- **Embeddings estáticos**: Modelos generalmente basados en aprendizaje supervisado o semi-supervisado que consiguen una representación vectorial fija de cada palabra asociada en el corpus. En esta corriente se ven los embeddings de las palabras como la "salida"de los modelos. Suelen entrenar sobre datasets muy grandes (como volcados de la Wikipedia por ejemplo) ya que en mayor o menor medida se aprovechan de correlaciones existentes entre las palabras ya que para poder ajustar su modelo necesitan mucha información.
  Sus máximos exponentes a día de hoy son *word2vec* (Mikolov et al. [2013]), *FastText* (Bojanowski et al. [2017]) y *GloVe* (Pennington et al. [2014]).
- **Embeddings contextuales**: Sistemas basados generalmente en aprendizaje semi-supervisado o no supervisado, se empezaron a utilizar *LSTMs* combinadas con bases de conocimiento externo hace unos años a raíz de los estudios que se realizaron al utilizar como nuevas características para las palabras los vectores asociados a *emmbeddings* estáticos (Iacobacci et al. [2016], Yuan et al. [2016]). Esto llevó al desarrollo de los Transformadores (Vaswani et al. [2017]) que finalmente condujo a los modelos que, a conocimiento del autor, dan los mejores resultados a día de hoy:
  - *BERT* y variaciones (*RoBERTa* (Liu et al. [2019]), *AlBERT* (Lan et al. [2019]))
  - *XLNet* (Yang et al. [2019]
  - *ElMo* (Peters et al. [2018]
  - *GPT,GPT-2* (Radford et al. [2018])

Se pueden observar los diferentes paradigmas en la Figura 2.1.

Los embeddings asociados a las palabras son la base de los modelos de desambiguación lingüística que se utilizan hoy en día (o bien directamente utilizando medidas basadas en la estadística (Iacobacci et al. [2016] o bien utilizando los vectores como representaciones intermedias de estados mientras la

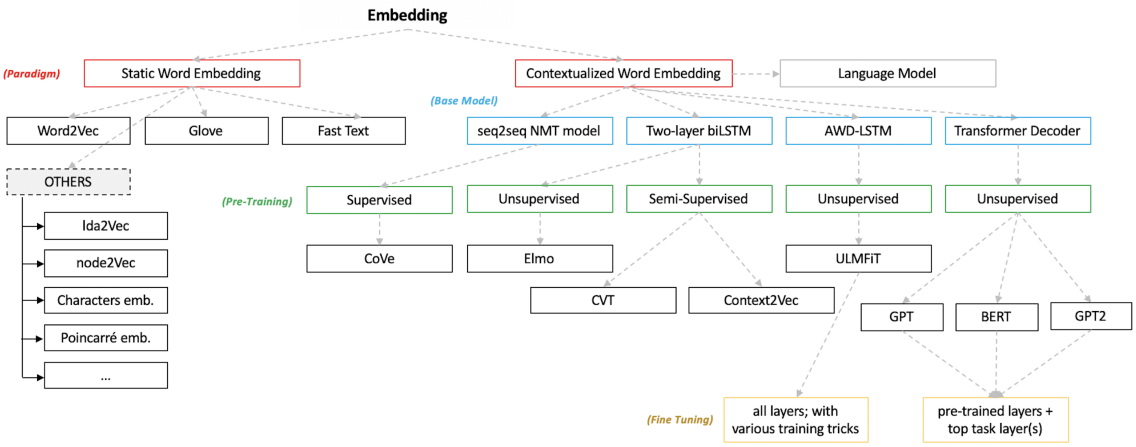


Figura 2.1: Ramas de modelos según embedding

red crea embeddings contextuales como *GlossBert* (Huang et al. [2019]), por eso no es de extrañar que paralelamente a las ramas en las que se clasifican los modelos según la forma de conseguir los embeddings, exista una estructura equivalente para clasificar los modelos dedicados a la desambiguación lingüística (Wiedemann et al. [2019]).

Antes de estudiar que modelo tiene asociados los mejores resultados, conviene hacer un inciso en que no todos los modelos se evalúan con el mismo dataset y en ocasiones puede ser realmente complicado decidir que modelo puede ser "mejor". Aun así dentro de la comunidad dedicada al tratamiento del problema de la desambiguación lingüísitica, se suelen considerar estándar el dataset relacionado con la competición SemEval del año 2007 *SE7* (Navigli et al. [2007]) (a veces también 2002,2003) y *GLUE* (Wang et al. [2018]).

Dicho esto, a conocimiento del autor los mejores resultados vienen dados por variaciones de BERT (GlossBert en particular), IMS+GloVe embeddings (Iacobacci et al. [2016]) y XLNet (mejor resultado GLUE).

También hay que destacar un factor relevante a la hora de utilizar estos modelos: el tiempo de entrenamiento. Los modelos nuevos (XLNet, BERT & variaciones, GPT-2) son modelos que se encuentran en la rama del aprendizaje semi-supervisado y como tal entrenan una parte sobre un dataset etiquetado (parte supervisada) y otra sobre corpus inmensos sin etiquetar (parte no supervisada). La parte supervisada no suele llevar mucho tiempo pero la parte no supervisada puede llegar a requerir dias utilizando la potencia de una GPU, se pueden ver datos del tiempo de entrenamiento para algunos modelos en la Tabla 2.1.

Los datos de la tabla Tabla 2.1 se obtienen de la equivalencia $5{,}3GPU \sim 1TPU$ (33bits) (GPU vs TPU), por lo que si BERT tarda 4 días en 16TPUs paralelas en la nube (Devlin et al. [2019]), entonces tardará estimadamente 340 días en GPU. El resto de datos se obtienen de model_time_scalers.

Model	GPU (no parll.) days	STS performance
BERT	340	90.0
RoBERTa	1530	92.2
XLNet	1700	91.6

Tabla 2.1: Comparación del tiempo de entrenamiento previo y el rendimeinto

Figura 2.2: Tendencias de búsqueda de los términos "word embeddingz "word2vec"desde 2004

3

Bosques contextuales

3.1. Motivación

Imaginemos por un momento, que se esta leyendo las noticias y de pronto aparece un titular:

'Tesla ha presentado esta mañana su nuevo hibrido. "

En ese mismo momento, el cerebro humano, en apenas unas décimas de segundo, ejecuta una operación maravillosa y se percata de que ese *"Tesla"* se refiere a la compañía norteamericana Tesla Inc. y ese *"híbrido"* se refiere a un tipo de automóvil con un motor de combustión interna y uno eléctrico. ¿Pero cómo puede saber esto el cerebro humano? La respuesta a esto está lejos de estar clara pero una de las teorías existentes afirma que el núcleo del proceso podría estar en la asociación por memoria (Duch et al. [2008]). Esto impulsa a pensar que si queremos que una máquina sea capaz de realizar estas operaciones debe de tener algo parecido a la memoria humana. Nótese que la memoria humana no son solo conceptos almacenados sino también las relaciones entre estos, y por ello se necesita una estructura con una red de interrelaciones entre sus unidades de conocimiento básicas.
Siguiendo este principio, en vez de tomar una base de datos de conocimiento externa fija como *WordNet* (Miller [1995]) podemos tomar como tal estructura la Wikipedia (siendo las unidades básicas las páginas y sus correlaciones los links que existen entre ellas) y aprovecharnos de una estructura en constante actualización.
Usando esta idea, imaginemos que tenemos dos palabras polisémicas en la misma frase, se puede construir un modelo que vaya buscando la existencia de una relación "fuerte" semánticamente entre posibles significados de estas palabras buscando recursivamente en las páginas asociadas a ellas.
Se puede ver la idea ilustrada en un ejemplo en la Figura 3.1

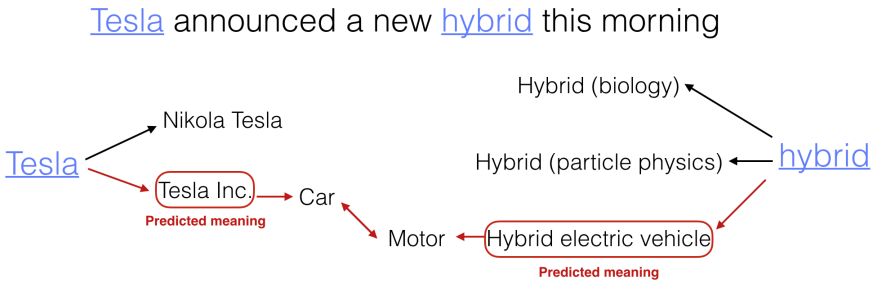


Figura 3.1:

3.2. Trabajos previos

No es, ni mucho menos, la primera vez que se utiliza Wikipedia como una fuente de conocimiento externo en problemas relacionados con el Procesamiento de Lenguaje Natural. Debido a la cantidad de artículos sobe una inmensidad de temas (actualmente más de 6 millones de páginas en la Wikipedia inglesa) ha probado ser un recurso muy valorado para aquellos que buscan mejorar los *embeddings* de su vocabulario (Camacho-Collados et al. [2015b],Camacho-Collados et al. [2015a]) o para los que aprovechan su estructura jerárquica para crear modelos basados en grafos (Agirre et al. [2015]). Sin embargo, en la aplicación propuesta en Camacho-Collados et al. [2015b] se utilizan a la hora de obtener un subcorpus para un concepto, las páginas que contienen a dicho concepto c y las páginas asociadas a los Synsets de *BabelNet* (Navigli and Ponzetto [2012]) asociadas a los hiperónimos de c. Sin embargo esto solo permite un razonamiento inductivo y Agirre et al. [2015] establece que utilizar el grafo de wikipedia como un grafo dirigido y no recíproco empeora el rendimiento del modelo y se tiene la misma situación en Camacho-Collados et al. [2015a].
En Agirre et al. [2015] se establece que solo se toman las conexiones "directas" entre páginas, y aunque es cierto que esto garantiza que los conceptos estarán relacionados, la expansión del grafo de esta forma limita mucho la información que de él se puede extraer; por ejemplo en la página *Cat* de la Wikipedia inglesa, no existe un enlace directo a la página *Dog* y estos dos conceptos tienen bastante probabilidad de estar relacionados en una frase.
Una aproximación al problema que también usa Wikipedia y un Framework basado en múltiples bases de conocimiento externo para crear un sistema de aprendizaje semi-supervisado se puede encontrar por ejemplo en Wang et al. [2020]. Sin embargo sus resultados en el SemEval07 quedan por detrás de modelos como GlossBERT o BERT y el Framework propuesto no está preparado para soportar múltiples idiomas.

A conocimiento ningún autor ha creado una herramienta de desambiguación independiente del idioma que no necesite corpus de entrenamiento.

3.3. SpaCy & NLTK

Para el desarrollo del modelo se han usado herramientas proporcionadas por las librerías de Procesamiento de Lenguaje Natural *SpaCy* (Honnibal and Montani [2017]) y *NLTK* (Elhadad [2010]). El modelo se vale de estas herramientas para crear un diccionario asociado a cada página de Wikipedia; este diccionario D estará formado por un conjunto de pares clave-valor en la que las entradas tienen forma:

$$D[(\textit{stem}(\text{p}), \textit{pos}(\text{p}))] = (W_p, O_p, R_p)$$

Donde p es una palabra de la página de Wikipedia y

- $stem(\cdot)$: es una función que devuelve la raíz de una palabra (en particular *Snowball Stemmer* de *NLTK*)
- $pos(\cdot)$: indica lacategoría gramatical de la palabra (en particular un es el atributo *Token.pos_* de *SpaCy*
- W_p: es el conjunto de palabras relacionadas con esta raíz (por ejemplo fábrica,fabricación,fábricas)
- O_p: son las posiciones de las palabras de W_p en el texto de la página de Wikipedia
- R_p: es la importancia que tiene la palabra en particular, se incializa a 0 y se asigna después (ver Sección 2.4)

Nota: Solo se añaden al diccionario las palabras con un significado léxico (Sustantivos,adverbios,adjetivos y verbos), pues son las que aportan mayormente el significado semántico a la frase.
El proceso de creación del diccionario se encuentra esquematizado en la Figura 3.2.

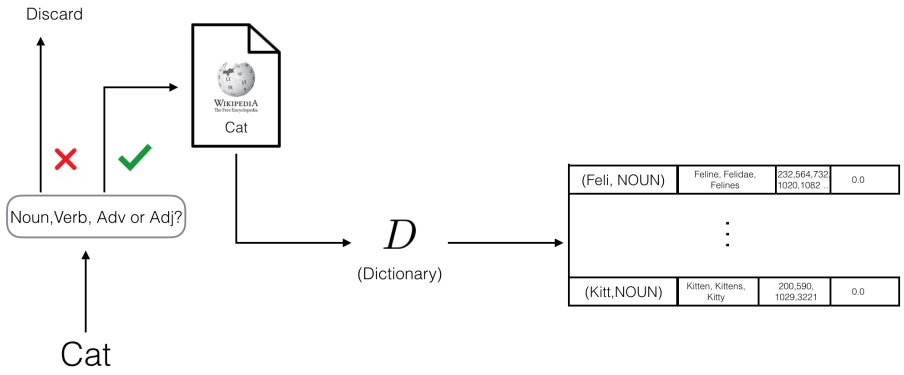


Figura 3.2: Proceso de creación de diccionario

Al obtener los sustantivos de una frase, es importante el conjunto de palabras que hacen el "papel" de sustantivo pueden estar formadas por más de una palabra, por ejemplo en la frase en inglés:

"The dog ate the cat food "

Sería deseable que nuestro modelo no solo reconociese "dog" como sustantivo sino que además reconociese *"the cat food"* como una sola entidad, esta tarea en el modelo la lleva acabo la funcionalidad *noun_chunks* proporcionada por SpaCy.

3.4. La importancia de una palabra en un texto

En la Sección 3.1 se mencionó la necesidad de encontrar relaciones "fuertes" entre dos conceptos. Vamos a definir formalmente la estructura que crea el algoritmo alrededor de un concepto; esta estructura nos ayudará posteriormente a definir lo fuerte que es el enlace entre dos conceptos.
Imaginemos que tenemos un universo de conceptos $\mathcal{U}$, sean dos conceptos $c_1, c_2 \in \mathcal{U}$, asociados a las páginas de Wikipedia $\mathcal{W}_1, \mathcal{W}_2$. En la sección anterior vimos que se construye un diccionario para cada una de estas palabras, sean esos diccionarios d_1, d_2, formalmente:

$$d_i \colon \mathcal{W}_{\mathcal{U}} \times \{\text{N,V,Ad,Adv}\} \longrightarrow \mathcal{U}^{k+1} \times \mathbb{N}^k \times [0,1]$$
$$\mathcal{W}_i \longmapsto d_i(\mathcal{W}_i)$$

En Agirre et al. [2015] se expone que al usar el hipergrafo de Wikipedia se debe de tener en cuenta como conceptos muy relacionados aquellos que tienen una conexión directa recíproca (i.e. $\mathcal{W}_i$ tiene un enlace a $\mathcal{W}_j$ y $\mathcal{W}_j$ tiene un enlace a $\mathcal{W}_i$) o de lo contrario afectará al rendimiento. Esto es esencialmente correcto pues como señalan los autores, cuando esto ocurre, debe de existir una relación entre los artículos. Sin embargo, esta técnica deja fuera muchas relaciones semánticas y es por ello por lo que es necesario flexibilizar esta definición sin caer en relacionar cosas que no lo estén. Para ello se da la siguiente definición:

Definición 2.1: *Sean $\mathcal{L}_1, \mathcal{L}_2$ los links de las páginas de Wikipedia $\mathcal{W}_1, \mathcal{W}_2$ respectivamente. Se define la* ***correlación*** *entre dos páginas como:*

$$d(\mathcal{W}_1, \mathcal{W}_2) = |\mathcal{L}_1 \cup \mathcal{L}_2| \sum_{l \in \mathcal{L}_1 \cup \mathcal{L}_2} \min\{I_{\mathcal{W}_1}(l), I_{\mathcal{W}_2}(l)\}$$

Donde $I_{\mathcal{W}_i}(\cdot)$ mide la importancia de un link en la página $\mathcal{W}_i$

Cuadro 3.1: Correlación entre dos páginas

Esta definición no está basada en enlaces directos sino en enlaces comunes. Viene motivada no solo por si existen o no enlaces entre dos páginas sino que también busca la calidad de esta unión (nótese como una alta correlación implica necesariamente páginas con muchos enlaces en común y/o con una importancia de enlaces muy fuerte en ambas páginas). Sin embargo, de esta definición surge naturalmente la pregunta de como medir la importancia de un enlace entre dos páginas y para ello primero es necesario calcular cual es la importancia de una palabra determinada en una página.
Una de las formas más populares de medir la importancia de una palabra en un texto es la métrica conocida como *TF-IDF* (Ramos [2013]), que utiliza la relación entre la frecuencia de la palabra relativa a todo el corpus y la frecuencia del propio texto para medir la relevancia de la palabra en ese texto

en particular. En el modelo no es posible usar esta métrica ya que la navegación por el grafo de la Wikipedia se realiza como una búsqueda dinámica sobre la propia web (ver la sección de Interfaz con Wikipedia). En lugar de *TF-IDF* se propone usar una métrica basada en la homogeneidad de distribución de una palabra:

Definición 2.2: *Sea* $\mathcal{W} = w_{ii\in\{1,..,n\}}$, *una página de Wikipedia con* n *palabras ordendas (no necesariamente distintas) y sea* $\mathcal{B}_k = w_i, ..., w_{i+k}$ *un bloque de* k *palabras contiguas. Definimos entonces la* ***importancia*** *de una palabra* $w_j \in \mathcal{W}$ *como:*

$$I_{\mathcal{W}}(w_j) = P(w_j \text{ aparezca en } \mathcal{B}_k \text{ genérico}) = P(w_j|\mathcal{B}_k)$$

Cuadro 3.2: Importancia de una palabra en una página

Lo primero que tenemos que notar es que esta medida no es igual a la medida de frecuencia de palabras (ver Figura 3.3)

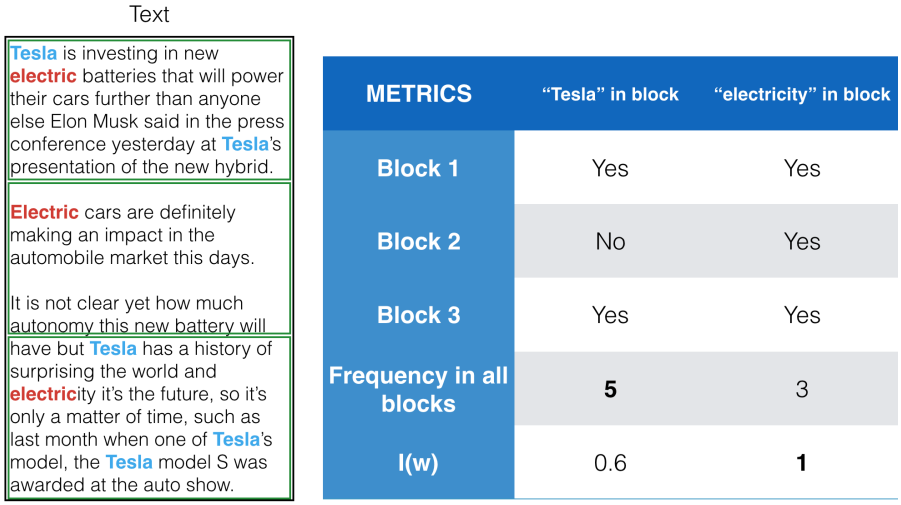


Figura 3.3: Comparación de la medida de frecuencia y la de importancia

Partiendo de la expresión de la medida de importancia tenemos que:

$$\begin{aligned} I_{\mathcal{W}}(w_j) &= P(w_j|\mathcal{B}_k) \\ &= \frac{P(w_j)P(\mathcal{B}_k|w_j)}{P(\mathcal{B}_k)} \\ &= \frac{|\{w \in \mathcal{W} : w_j = w\}|}{|\mathcal{W}|} \cdot \frac{1}{|\mathcal{W}| - k + 1} \cdot P(\mathcal{B}_k|w_j) \end{aligned}$$

Ahora bien, tenemos que $\exists \beta \in \mathbb{R}$:

$$P(\mathcal{B}_k|w_j) = P(\mathcal{B}_1|w_j)^\beta = P(w_j) = \frac{|\{w \in \mathcal{W} : w_j = w\}|}{|\mathcal{W}|}$$

Por tanto podemos aproximar la distribución probabilidad de $I_\mathcal{W}(w_j)$ como:

$$I_\mathcal{W}(w_j) \sim P(w_j)^\alpha \text{ con } \alpha \in \mathbb{R} \tag{3.1}$$

Es por ello necesario estudiar cual es la distribución de probabilidad de encontrar una palabra en un texto, o lo que es lo mismo, estudiar su frecuencia. Zipf [1950] postuló que la distribución de la frecuencia de una palabra en un texto sigue una *Ley potencial*, es decir, que la segunda palabra más frecuente aparece la mitad de veces que la primera, la tercera un tercio, y así sucesivamente. Podemos expresar esto formalmente si entendemos que w^n es la n-ésima palabra más frecuente como:

$$freq(w^n) = \frac{freq(w^1)}{n^\alpha}, \quad \alpha \in \mathbb{R}$$

Utilizando este hecho, de la Ecuación 3.1 se obtiene que:

$$I_\mathcal{W}(w_j) \sim X^\alpha \text{ con } \alpha \in \mathbb{R}, \quad X \sim Zipf \tag{3.2}$$

Se pueden comprobar estos resultados tomando una página de Wikipedia cualquiera, ordenando las palabras por importancia y contrastándolas con un modelo de Zipf elevado a una constante real. Ajustado por mínimos cuadrados se muestra un Error Cuadrático Medio en la predicción del orden de 10^{-4} (ver Figura 3.4).

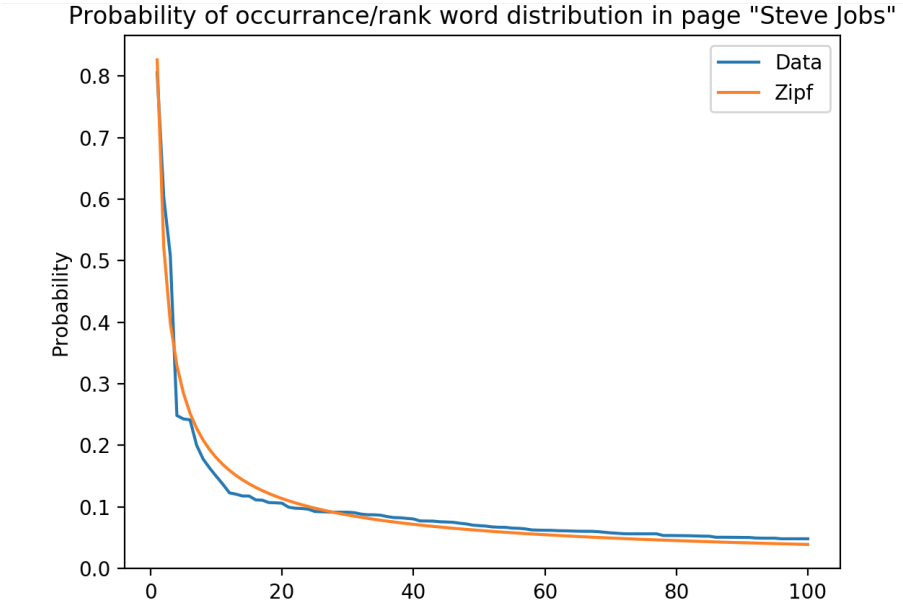


Figura 3.4: Ajuste de la importancia por modelo de Zipf

De esta forma se puede asociar un modelo ajustado de Zipf a cada página de Wikipedia, de ahora en adelante nos referiremos a este modelo como $\mathcal{Z}_\mathcal{W}$.

Aunque el cálculo de $I_{\mathcal{W}}(w_j)$ se hace de manera efectiva con un bucle y conteo sobre las palabras de $\mathcal{W}$, es muy útil conocer cómo se distribuyen las palabras en el texto, ya que esto nos proporciona una forma de medir la relevancia de un enlace:

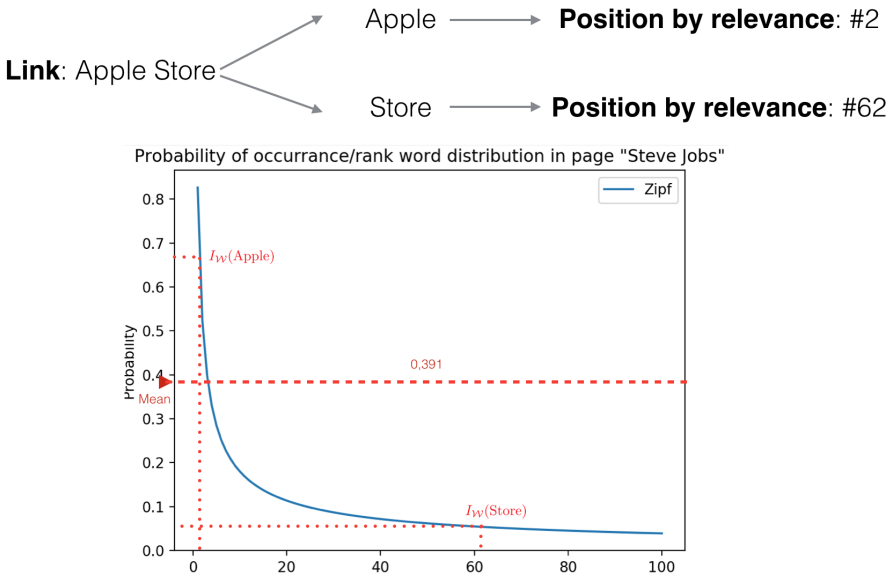
Definición 2.3: *Sea $\ell \subset \mathcal{W}$ un link de la página de Wikipedia. Entonces definimos la importancia del link como:*

$$I_{\mathcal{W}}(\ell) = \mathcal{Z}_{\mathcal{W}}(\frac{1}{|\ell|}\sum_{w^i \in \ell} i)$$

Donde como antes w^i corresponde a la i-ésima palabra más importante en la página $\mathcal{W}_i$.

Cuadro 3.3: Importancia de un link

Un ejemplo del calculo de la importancia de enlace así definida frente a la técnica de hacer la media de las palabras que lo conforman se puede observar en la Figura 2.5.



$$I_{\mathcal{W}}(\text{Apple Store}) = \mathcal{Z}_{\mathcal{W}}(\frac{1+62}{2}) = \mathcal{Z}_{\mathcal{W}}(31.5) = 0.113$$

Figura 3.5: Comparación entre la imagen de la media y la media de la imagen

Nótese que la diferencia entre tomar la media o no es un factor de 3 en este caso, la idea de no tomar directamente la media viene motivada por el comportamiento de la distribución de Zipf. Si se tienen dos palabras $w^i, w^j \in \mathcal{W}$ hacer la media directa de su imagen resulta en $\frac{\mathcal{Z}(1)(i+j)}{ij}$, que tiene un crecimiento muy rápido para valores de i, j pequeños y muy lento para valores de i, j grandes frente al crecimiento

más suave que proporciona la imagen de la media.

3.5. Búsqueda en Wikipedia

Como se ha dejado ver en las secciones anteriores, una de las partes centrales del modelo consiste en el conocimiento externo que extrae de Wikipedia. A diferencia de otros autores (Camacho-Collados et al. [2015a], Iacobacci et al. [2016], Agirre et al. [2015], Wang et al. [2020]) y como ya se comentó antes, el corpus de la Wikipedia no se descarga en ningún momento sino que se busca el conocimiento de manera jerárquica en el grafo formado por sus páginas y enlaces.
Para estructurar las búsquedas distinguimos entre tres tipos de páginas dentro de la Wikipedia:

- Página principal: Una página de Wikipedia de un concepto concreto y sin ambigüedad, por ejemplo "*Apple Inc.*"
- Páginas internas de Wikipedia: Aquí se incluyen las categorías (que se ha probado que no ofrecen una diferencia significativa a la hora de realizar predicciones (Camacho-Collados et al. [2015b]) y otras como las listas o los índices.
- Páginas *disambiguation*: Un tipo de páginas especiales con todas las páginas principales posibles asociadas a una palabra.

Existen numerosas librerías en Python para realizar peticiones a Wikipedia; durante el desarrollo del modelo se optó por WikipediaApi (Majlis [2017]) por su rapidez y su facilidad de uso. Nótese que soporta múltiples lenguajes y por tanto junto con SpaCy hace que el modelo se pueda utilizar independientemente del lenguaje siempre que sea aceptado por ambas librerías.
Durante las anteriores secciones se ha hecho referencia a la página de Wikipedia asociada a un concepto. En realidad, en la práctica esto es ligeramente más complicado. Un concepto puede o no tener varias páginas principales asociadas dependiendo de la especificidad de este. Por ejemplo *Apple* como concepto puede referirse a una treintena de cosas distintas y como expusimos antes *Apple Inc.* solo puede referirse a una. Sin embargo contamos con una ventaja y es que si existe más de una página principal asociada a un concepto, entonces existe su página de disambiguación, y los enlaces de esta son todas las páginas principales a las que el concepto puede estar asociado (acepciones). Se puede ver un ejemplo de búsqueda en el Anexo C.
La búsqueda sobre el grafo de Wikipedia se hace considerándolo como un grafo no dirigido para poder aplicar razonamiento tanto inductivo como deductivo.
Buscar jerárquicamente por la Wikipedia de esta forma, requiere en general configurar previamente dos parámetros que modelan la búsqueda: la profundidad a la que se puede llegar en la búsqueda y la regularidad de la expansión i.e. cuantos enlaces se pueden tomar para buscar recursivamente en cada página. Esto sugiere que vamos a necesitar una heurística para poder controlar que el número de nodos no crezca hasta límites impracticables computacionalmente hablando como se ve en la Figura 3.6.

La conclusión que se obtiene de estos resultados es que el crecimiento del número de nodos a explorar es tan grande que sugiere que cualquier tipo de algoritmo de búsqueda práctico tendrá que

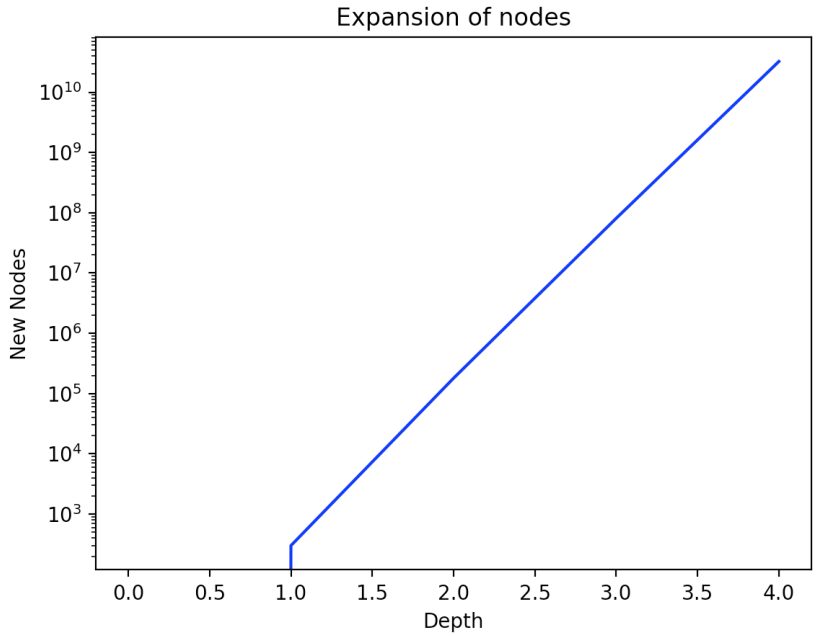


Figura 3.6: Expansión de nodos sin control de regularidad

estar basado en una heurística y la expansión, basada en regularidad o en algún tipo de poda como se verá durante las próximas secciones.

3.6. Nodos y Árboles

Las secciones anteriores motivan la idea de que si se va a usar conocimiento extraído dinámicamente del grafo de Wikipedia, una buena forma de aprovechar la estructura jerárquica que esta proporciona es utilizar árboles como estructura de datos principal del algoritmo.
Para ello habrá que definir que información está asociada con un nodo y cual es la forma en la que se crean los árboles a partir de un nodo raíz dado.
Entenderemos un nodo como una asociación entre un concepto y una página de Wikipedia. Un nodo como estructura de datos contendrá:

- **title** (*string*): concepto abstracto asociado al nodo raíz (el más antiguo de sus progenitores).
- **page** (*WikipediaApi.Wikipedia.Page*): la página de Wikipedia asociada con el concepto del nodo.
- **dictionary** (*dict*): diccionario de lexemas de la página de Wikipedia asociada (estructura del diccionario explicada en la Sección 3.3.).
- **model** (*function*): modelo de Zipf ajustado a los valores de relevancia de las palabras (ver Sección 3.4.
- **parent** (*Node*): Enlace a nodo padre.
- **children** (*dict*): Diccionario con los nodos hijo; las claves son de tipo *Node* y los valores son *float* y representan el peso de enlace entre dos nodos.
- d (*int*): regularidad de expansión, se expanden los d hijos con enlaces de mayor peso.
- depth: profundidad a la que se encuentra el nodo con respecto al más antiguo de sus progenitores (raíz)

En la definición de nodo como estructura de datos ya se ha dejado entrever que uno de los aspectos claves va a ser la expansión de nodos.
La expansión de un nodo consiste en tomar los d enlaces más relevantes de un nodo y fijar como nodos hijos los nodos asociados a estos enlaces. Nótese que los enlaces representan una conexión a una página de Wikipedia directa, no pueden tener ambigüedad alguna y cada uno tiene un peso de enlace (su importancia) asociado.
Al expandir nodos obtenemos nodos relacionados con conceptos de manera inequívoca pero esto no es así si el nodo que se expande es la raíz, en este caso, los hijos corresponderán a **todos** los enlaces de la página de *disambiguation* asociada al concepto que se intenta desambiguar y los pesos de enlace serán todos iguales (probabilidad uniforme) representando que a priori ningún enlace tiene más probabilidad de otro de ser el asociado al concepto que intentamos desambiguar. Se puede ver un ejemplo ilustrado de expansión en la Figura 3.7.

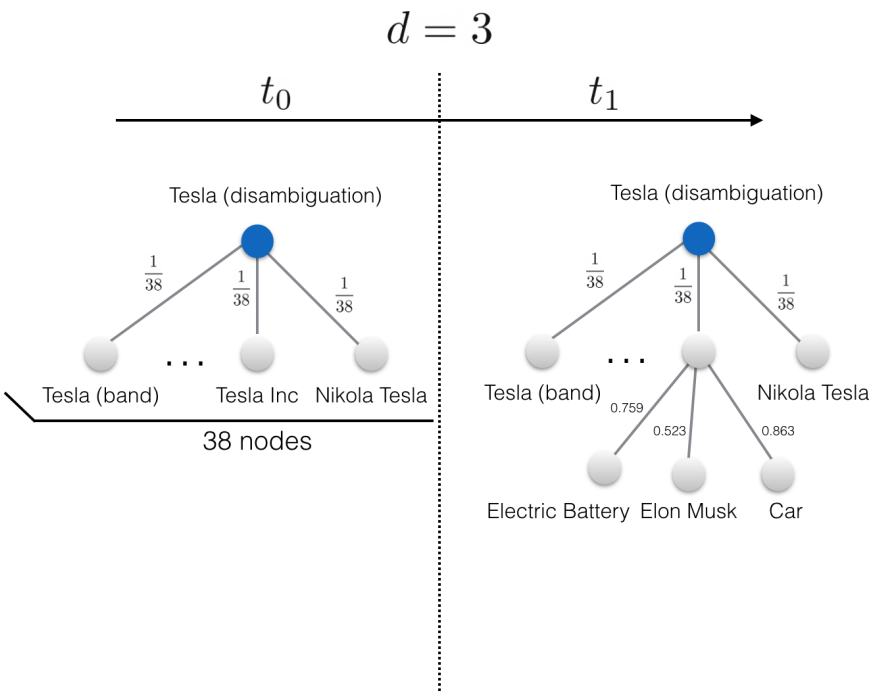


Figura 3.7: Representación gráfica de la expansión de dos nodos

3.7. Creación del bosque

En la sección anterior se ha mostrado como se puede crear un árbol a partir de un nodo raíz asociado a un concepto que puede o no tener varios artículos de Wikipedia asociados pero ¿Cómo se llega a la desambiguación de conceptos? Para ello, empecemos considerando solo dos árboles $\mathcal{T}_1$

y $\mathcal{T}_2$ que vienen de dos conceptos c_1, c_2 en los que al menos uno de ellos tiene varias acepciones asociadas.

En cuanto se empiece a expandir estos árboles, tarde o temprano encontraremos un nodo en común entre ambos, y esto proporcionará una relación entre c_1, c_2 en forma de camino que a su vez asociará una de las posibles acepciones a los conceptos polisémicos. Pero ¿Cómo podemos estar seguros de que este es realmente el camino que lleva a las acepciones correctas? ¿Y en que orden se exapanden los nodos de un árbol?

La respuesta a estas dos preguntas viene dada por una respuesta común de la que ya se tenía una pista despues de los resultados de expansión vistos en la Sección 2.5: Se necesita una heurística.

Hasta ahora se ha definido una forma de calcular la relevancia de una palabra y de un enlace en una página de Wikipedia, pero se va a necesitar también una forma de medir cual es la relación entre dos nodos que a priori no están conectados directamente.

Definición 2.4: *Sean n_1, n_2 dos nodos como se ha definido arriba y sean $\mathcal{L}_1, \mathcal{L}_2$ los links de sus páginas asociadas $\mathcal{W}_1, \mathcal{W}_2$. Entonces en primer lugar se define su δ-intersección como:*

$$\mathcal{L}_1 \cap_\delta \mathcal{L}_2 = \{\ell \in \mathcal{L}_1 \cap_\delta \mathcal{L}_2 : I_{\mathcal{W}_1}(\ell) \geq \delta \text{ y } I_{\mathcal{W}_2}(\ell) \geq \delta\}$$

En estas condiciones se define la similitud entre dos nodos como

$$I(n_1, n_2) = \min_{\ell \in \mathcal{L}_1 \cap_\delta \mathcal{L}_2} \{I_{\mathcal{W}_1}(\ell), I_{\mathcal{W}_2}(\ell)\}$$

Cuadro 3.4: Similitud entre dos nodos

El parámetro δ se puede fijar al valor que se desee siempre cumpliendo $\delta \in [0, 1)$ y teniendo en cuenta que cuanto mayor sea más restrictivo será el conjunto.

Utilizando la similitud entre dos nodos como una heurísitca para la expansión de nodos tenemos que en cada iteración se expanden los pares de nodos provenientes de distinta raíz con mayor similitud y se para el algoritmo cuando se encuentra un camino entre las dos raíces.

Desde este punto generalizar a un algoritmo más general es sencillo: Dado un conjunto de conceptos $\mathcal{C} = \{c_1, c_2..., c_n\}$ se utiliza la técnica de la expansión por pares de nodos de raíces distintas y se van desambiguando las raíces hasta tener una componente conexa que las conecte a todas. Se puede ver un ejemplo ilustrativo en la Figura 3.8 y un ejemplo de salida en Figura 3.9.

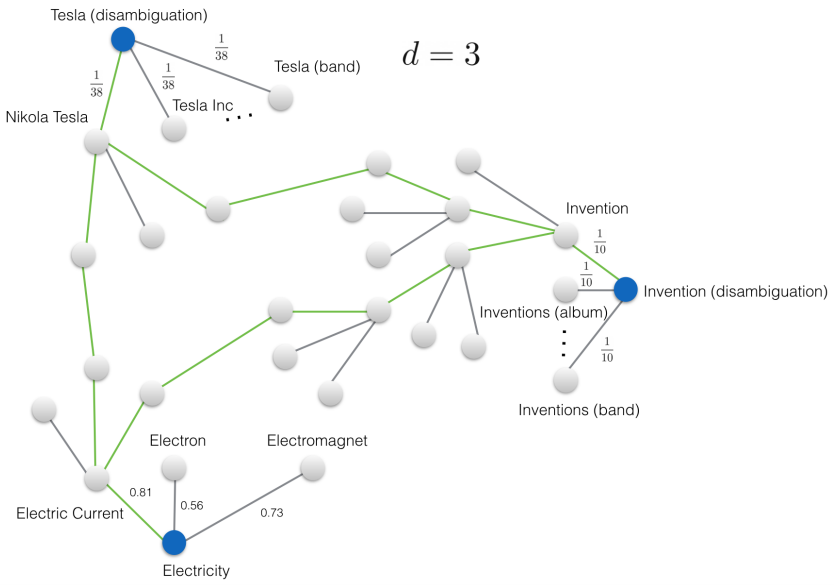


Figura 3.8: Representación gráfica de un bosque contextual

```
fr = algorithm("Apple and Microsoft have always competed in tech")
```

```
{Apple: 'apple', Microsoft: 'microsoft', tech: 'tech'}
['Apple', 'Microsoft', 'tech']
Word:Apple
Word:Microsoft
Word:tech
Starting Queue...
Combinations 3
Starting Expansion...
Selected nodes Apple Inc.,Microsoft
Conexion between trees apple,microsoft established
Starting Expansion...
Selected nodes Apple,Technology
Conexion not found, expanding new nodes...
Starting Expansion...
Selected nodes Asia,Science
Conexion not found, expanding new nodes...
Starting Expansion...
Selected nodes Continent,Nature
Conexion between trees apple,tech established
Starting Expansion...
Selected nodes Microsoft,Technology
Conexion not found, expanding new nodes...
Starting Expansion...
Selected nodes Software development,Computer science
Conexion between trees microsoft,tech established
```

```
In [70]: print(fr.dic["apple"].page.text[:100])
         print(fr.dic["microsoft"].page.text[:100])
         print(fr.dic["tech"].page.text[:100])
```

```
Apple Inc. is an American multinational technology company headquartered in Cupertino, California, t
Microsoft Corporation () is an American multinational technology company with headquarters in Redmon
Technology ("science of craft", from Greek Τέχνη, techne, "art, skill, cunning of hand"; and -λογία,
```

Figura 3.9: Ejemplo de utilización del algoritmo

3.8. Limitaciones

El modelo de bosques contextuales es ineficiente para textos largos en su versión actual; esto se debe a que en textos largos se puede ir almacenando contexto del texto leído, y por tanto sería ineficiente calcular bosques contextuales para cada frase por separado cuando se tiene una gran cantidad de contexto previo.

Actualmente la única implementación que existe del modelo está escrita en Python que es un lenguaje interpretado y por lo tanto lento comparativamente a los lenguajes no interpretados; además todo el código (por cuestiones de tiempo) está sin paralelizar, por esta razón ha sido imposible probar con uno de los dataset estándar de SemEval.

Es posible que el modelo de bosques contextuales tal y como está diseñado tenga problemas en lo referente al tiempo de ejecución con frases muy largas, ya que es relativamente común que un término tenga varias acepciones y al tomarse las combinaciones de nodos por pares para medir la heurística en la expansión, si una frase es demasiado larga el primer paso de expansión, en el que se toman todos los links de la página de desambiguación podría convertirse en un cuello de botella.

El modelo de bosques contextuales no descarga ningún corpus sino que ejecuta búsquedas jerárquicamente en Wikipedia para obtener conocimiento cuando lo necesita, esto hace al modelo dependiente de una conexión a Internet. Además las querys que hace a Wikipedia, aunque sean bastante ligeras, son muchas por lo que si la velocidad de la conexión a Internet se ralentiza, también lo hará el algoritmo.

4

Experimentos y resultados

En este capítulo se exponen los experimentos llevados a cabo para comparar el rendimiento del modelo en una tarea de desambiguación con los rendimientos obtenidos por *BERT* y *GloVe*. Se han elegido estos dos modelos como representantes de cada una de las ramas en las que se clasifican los modelos según la forma de realizar *embeddings* (ver Sección 1.2.).
Ambos experimentos se van a realizar sobre un dataset de 75 frases cortas en lengua inglesa y disponibles en el Anexo A. Cada una de las frases tiene marcada una palabra clave que es el objetivo a desambiguar.
Por motivos de tiempo no ha sido posible implementar el modelo para utilice verbos y adjetivos en la desambiguación, por lo que para comparar realmente el rendimiento de los modelos, la información semántica necesaria para desambiguar la frase en cada una de las 75 frases del dataset se encuentra en otro sustantivo de la frase (que puede ser ambiguo o no).
Se considera una predicción correcta si en la predicción que se hace de la página de Wikipedia asociada a un concepto, ese concepto se usa en el mismo contexto. Por ejemplo si la frase es *"Apple and Microsoft are computer companies"* entonces se consideran correctas como predicciones de *"Apple"*: Apple TV, Apple Inc. Apple Store...

4.1. BERT

BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) (Devlin et al. [2019]) es un modelo NLP de representación de lenguaje desarrollado por Google en 2018 con el fin de mejorar la comprensión de las búsquedas realizadas por los usuarios. Se trata de un modelo basado en representaciones vectoriales contextuales de frases usando pre-entrenamiento y ayudandose del aprendizaje semi-supervisado de secuencia y los transformadores para conseguir resultados que marcaron (y sus variaciones siguen marcando) un récord en el entendimiento de Lenguaje Natural.
La utilización de LSTMs de dos capas ya se había utilizado antes en modelos como las primeras versiones de *UMLFit* y *ELMo*. La revolución de *BERT* fue usar una nueva tecnología conocida como *Transformer* (Vaswani et al. [2017]) una arquitectura basada en el paradigma *Encoder-Decoder* que se

aprovecha de una nueva forma de aprendizaje conocida como *Seq2Seq* para representación iterativa de vectores (asociados en este caso a una frase). De esta forma, la arquitectura *Transformer* es una pieza fundamental de *BERT*, ya que así procesa las palabras en el contexto de la oración al completo, en lugar de separar individualmente cada palabra. De esta manera, el transformador es la clave para comprender la intención detrás de las búsquedas de los usuarios.

Sin embargo, lo que hace realmente único a *BERT* es su característica bidireccional, ya que no lee de manera secuencial, sino que lee la secuencia de palabras en dos direcciones, permitiendo que se pueda aprender de las relaciones contextuales entre palabras y de la temática en función de todo el entorno.

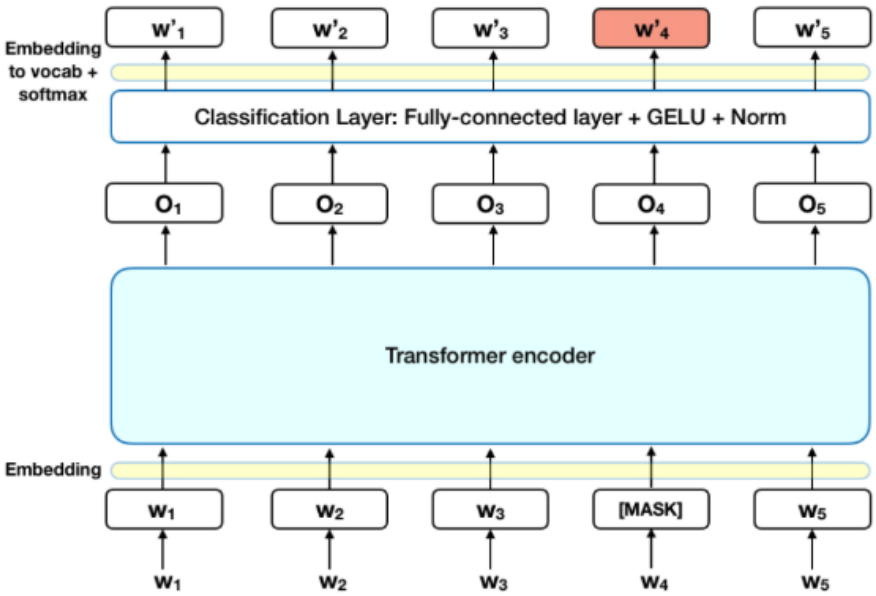


Figura 4.1: Arquitectura de Transformer basada en Seq2Seq de BERT

A pesar de la gran precisión de *BERT* y de que se trata de un hito en el Procesamiento de Lenguaje Natural, es más eficaz en una determinada clase de problemas, como por ejemplo los de similitud semántica de frases (ya que se diseñó inicialmente para la interpretación semántica de búsquedas).

Se va a aprovechar esta característica de *BERT* para poder adaptar su funcionamiento al problema de desambiguación de una palabra. En primer lugar se va a modificar la última etapa de funcionamiento interno ya que *BERT* fue originalmente diseñado para devolver el embedding de una frase, no el de las palabras por separado. Sin embargo, si en las últimas etapas del aprendizaje *Seq2Seq* en el transformador se toman los últimos 4 estados ocultos asociados a la palabra clave que queremos desambiguar, antes de que se pase a la red, podemos obtener 4 vectores asociados al embedding contextual de la palabra antes de que se utilice para obtener el embedding de la frase, haciendo la media podemos obtener un embedding contextual asociado a la palabra.

Una vez conseguida una forma de obtener un embedding contextual de una sola palabra en *BERT*,

es necesario obtener un conjunto de frases en la que la palabra clave que queremos desambiguar se use en cada una con una acepción diferente. Una vez más usaremos la página de *disambiguation* de la Wikipedia para obtener todos los artículos asociados con las acepciones de la palabra clave. Obteniendo la primera frase de cada uno de estos artículos ya tenemos el conjunto de frases buscado. Finalmente para decidir cual es la acepción que se está utilizando en la frase original, se comparan usando la distancia coseno el vector v asociado a la palabra clave en la frase original con cada uno de los vectores $d_1, ...d_k$ obtenidos de cada una de las frases recolectadas de los artículos de Wikipedia. En la Figura 3.2 se puede observar una representación gráfica del experimento.

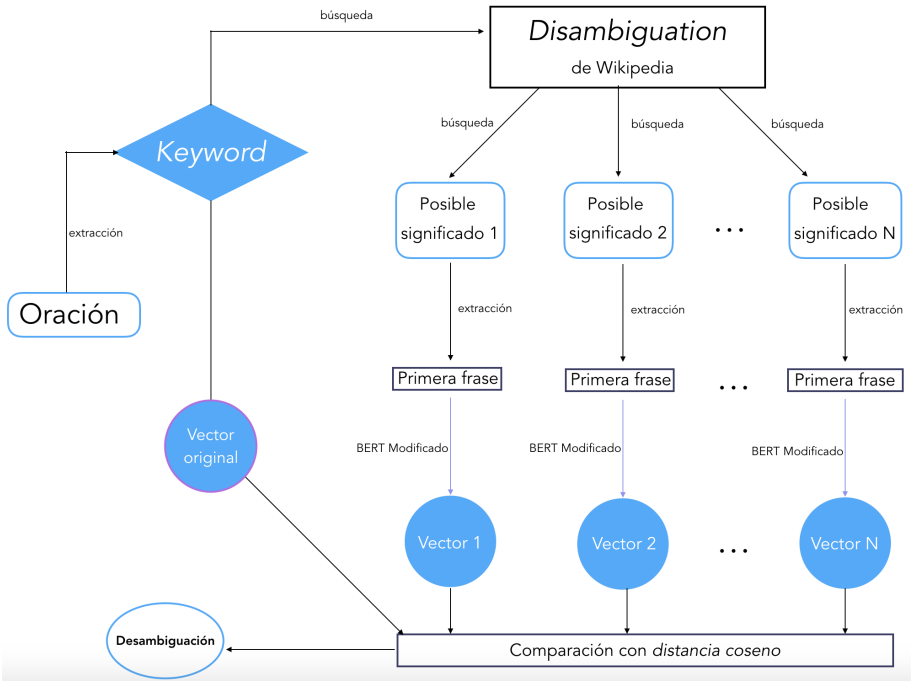


Figura 4.2: Representación gráfica del experimento BERT

Tras la ejecución del experimento para cada una de las 75 frases se obtiene que el acierto (ACC) obtenido por BERT es del 44.0 %.

Cuadro 4.1: Resultado del experimento de BERT

El resultado puede resultar extraño si se tiene en cuenta que *BERT* ha puntuado puntuaciones del 86 % de acierto en datasets estándar de comprensión lingüistica como SemEval (Navigli et al. [2007]). Cabe destacar que al estudiar las predicciones realizadas por *BERT* se observa que más de un 50 % de los casos en los que falla la predicción que hace es una entidad con nombre (una canción, una empresa...) esto se puede explicar porque el modelo básico de *BERT* (bert-base-uncased) entrenó

sobre un Dataset de texto en inglés en minúsculas en el que es probable que no hubiese muchas entidades. Además el dataset de SemEval tiene muchas palabras para obtener una representación lingüísitica, pero generalmente las frases tienen pocas palabras con un gran número de acepciones como ocurre en este caso lo que también explicaría la bajada de rendimiento. Es notable también que aunque el modelo falle intentando hacer la predicción exacta, en un 60 % de las frases la predicción está relacionada con el contexto (Ciencias,Música...)

4.2. GloVe

GloVe (**Gl**obal **Ve**ctors for Word Representation) es un algoritmo de aprendizaje no supervisado desarrollado en Stanford (Pennington et al. [2014]) que utiliza las ideas de probabilidad de co-ocurrencia de palabras en un corpus lingüístico grande (como la Wikipedia o el corpus de Brown) para crear representaciones vectoriales estáticas de palabras. Evidentemente este tipo de algoritmos, al dar embeddings estáticos ofrecerá siempre peores resultados que métodos que utilicen embeddings contextuales ya que es difícil llegar a un algoritmo que proporcione buenos resultados en la desambiguación lingüisitica por el choque que existe entre las (prácticamente infinitas) posibilidades de contexto frente a embeddings fijos para todas las palabras. Sin embargo, esta familia de métodos ofrecen además de los embeddings en sí, garantías de que se pueden realizar operaciones con los vectores obtenidos semánticamente hablando. Por ejemplo el vector más cercano que existe a *rey-hombre+mujer* será siempre *reina*. Además ofrecen la posibilidad de decir cuales son los vectores más cercanos a uno dado en el caso de que no exista desambiguación lo que puede ayudar a otras tareas como obtener las ideas clave de un texto.
Para este experimento, las frases con las distintas acepciones se obtienen de la misma forma que se hizo con *BERT* en el experimento anterior. Sin embargo hay que proponer una nueva forma de medir la similitud entre dos frases, pues no se puede comparar el vector asociado a la palabra clave a desambiguar (es siempre el mismo y tendría correlación absoluta con todas las páginas de desambiguación). Se propone entonces la siguiente métrica para comparar el contexto de dos frases:

Definición 3.1: *Sean* $s_1 = \{w_1^1, w_2^1, ..k, ..w_n^1\}, s_2 = \{w_1^2, w_2^2, .., k, ..w_n^2\}$ *dos frases con* n *y* m *vectores asociados a palabras que proporcionen información (NOUN,ADJ,ADV,VERB) y* k *el vector de la palabra clave a desambiguar. Entonces se define la distancia usando GloVe como:*

$$d_{GloVe}(s_1, s_2) = \frac{1}{mn - 1} \sum_{i,j} (1 - \cos(w_i^1, w_j^2))$$

Cuadro 4.2: Similitud entre dos frases con GloVe

Se va a entrenar el modelo con los embeddings obtenidos de una captura de la WIkipedia inglesa de 2014, son 6 billones de vectores de 300 dimensiones cada uno (http://nlp.stanford.edu/data/glove.6B.zip).

Tras la ejecución del experimento para cada una de las 75 frases se obtiene que el acierto (ACC) obtenido por GloVe es del 24.0 %.

Cuadro 4.3: Resultado del experimento de GLOVE

Era de esperar un acierto significativamente menor que el de *BERT* por usar embeddings estáticos. Este hecho ha llevado al modelo a predecir la misma salida para las distintas frases de una misma palabra,en un 36 % de los casos. Sin embargo cabe destacar la velocidad de cálculo frente a *BERT* (en particular BERT tarda 11.2 veces más tiempo que GloVe).

4.3. Comparación

El modelo del bosque contextual desarrollado está preparado para ejecutar la prueba con el dataset de test sin ninguna modificación de su estructura ni código extra necesario.

*Tras la ejecución del experimento para cada una de las 75 frases se obtiene que el acierto (ACC) obtenido por el modelo de bosque contextual es del **70.42** %.*

Cuadro 4.4: Resultado utilizando bosque contextual

Esto implica que el modelo obtiene un resultado 26 % mejor que el modelo básico de *BERT* y un 46 % mejor que *GloVe*. La mejora de resultados podría deberse a que el modelo de bosque contextual no se basa tanto en encontrar similitud de sentido semántico sino en buscar un contexto que desambigüe todas las palabras con más de una acepción de la frase, consiguiendo así un entendimiento profundo del significado que tiene una palabra en una frase.
Explorando los resultados fallidos del modelo, la mayoría de las frases en las que falla son frases en las que el verbo juega un mínimo papel en la disambiguación o ayuda al sustantivo base a desambiguar la palabra clave. Esto puede deberse a que el modelo todavía no está preparado para trabajar con verbos y cualquier mínima dependencia del contexto en ellos afectan a la calidad de su predicción.

5

Conclusiones y trabajo futuro

5.1. Conclusiones

En este trabajo se han definido los árboles contextuales, un modelo probabilístico basado en grafos para la desambiguación de palabras polisémicas en una frase que sin necesidad de conjunto de entrenamiento, ha probado tener mejor rendimiento que BERT (+ 26 %) y que GloVe (+ 46 %) en un experimento de desambiguación de 75 frases cortas con palabras polisémicas en inglés.
Esto prueba no solo que esta línea de investigación puede competir con los modelos preentrenados sino que además al no necesitar entrenamiento previo para poder operar, se puede cambiar el idioma en cualquier momento y seguir funcionando sin problemas. Esto abre nuevas posibilidades a la hora de crear nuevos modelos para el problema de desambiguación.

5.2. Trabajo futuro

El tener tan buenos resultados con respecto a los modelos que representan el estado del arte ahora mismo en desambiguación de sustantivos podría indicar que una vez que se amplíe el modelo para tomar en cuenta otro tipo de categorías gramaticales, como verbos o adjetivos, los resultados podrían competir con los últimos modelos pero sin ser necesario un pre-entrenamiento de días de duración.
También es necesaria una herramienta para reconocer estructuras gramaticales más complejas, para lo cual se está creando una librería que opera sobre SpaCy para poder reconocer patrones lingüísticos como por ejemplo los *phrasal verbs* en inglés ya que SpaCy por defecto no tiene esta opción (se adjunta el código de esta librería en el Anexo B).
Una vez se haya conseguido esta herramienta y se haya ampliado y optimizado el modelo (reprogramar en otro lenguaje como Julia o paralelizar y sacar las partes críticas a módulos de C), merecería la pena estudiar enfoques híbridos con modelos como GPT-2 o RoBERTa para la aplicación a otras tareas de procesamiento de lenguaje natural como Sentiment Analyisis aplicado a detección de noticias falsas (esto se investigará con más detalle como trabajo de Fin de Máster el año que viene) .
También merece la pena investigar como funcionarían embeddings hechos con el modelo de bosques

contextuales ya que el modelo respeta la estructura de poder decir cuales son los nodos más “cercanos” a uno dado sin perder la idea de contexto de los conceptos extraídos de una frase y de esto podrían beneficiarse otras tareas relacionadas con el Lenguaje Natural como la similitud semántica entre frases por asociación. Por ejemplo dadas las frases "Trump speaks to the media in Illinoisz "The president greets the press in Chicago"ser capaz de asociar el embedding de "president.ªl de "trump".
En el desarrollo del modelo de bosques contextuales se ha utilizado solo la Wikipedia como fuente de conocimiento externo, sin embargo existen muchas otras como *WordNet* (Miller [1995] o *BabelNet* (Navigli and Ponzetto [2012]), ¿Puede beneficiarse el modelo de extraer conocimiento de múltiples fuentes?
Finalmente, también se pueden explorar también otras métricas basadas en la distribución de Zipf, por ejemplo la distancia entre dos nodos podría ser la distancia (divergencia) de Jensen-Shannon.

Bibliografía

E. Agirre, A. Barrena, and A. Soroa. Studying the wikipedia hyperlink graph for relatedness and disambiguation, 2015.

P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. ISSN 2307-387X.

J. Camacho-Collados, M. T. Pilehvar, and R. Navigli. NASARI: a novel approach to a semantically-aware representation of items. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 567–577, Denver, Colorado, May–June 2015a. Association for Computational Linguistics. doi: 10.3115/v1/N15-1059. URL https://www.aclweb.org/anthology/N15-1059.

J. Camacho-Collados, M. T. Pilehvar, and R. Navigli. A unified multilingual semantic representation of concepts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 741–751, Beijing, China, July 2015b. Association for Computational Linguistics. doi: 10.3115/v1/P15-1072. URL https://www.aclweb.org/anthology/P15-1072.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

W. Duch, P. Matykiewicz, and J. Pestian. Neurolinguistic Approach to Natural Language Processing with Applications to Medical Text Analysis. *Neural networks : the official journal of the International Neural Network Society*, 21(10):1500–1510, Dec. 2008. ISSN 0893-6080. doi: 10.1016/j.neunet.2008.05.008. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2633093/.

P. Edmonds and E. Agirre. Word sense disambiguation. *Scholarpedia*, 3(7):4358, 2008. doi: 10.4249/scholarpedia.4358. revision #90370.

M. Elhadad. Natural language processing with python steven bird, ewan klein, and edward loper (university of melbourne, university of edinburgh, and bbn technologies) sebastopol, ca: O'reilly media, 2009, xx+482 pp; paperbound, isbn 978-0-596-51649-9; on-line free of charge at nltk.org/book. *Computational Linguistics*, 36:767–771, 12 2010. doi: 10.1162/coli_r_00022.

D. Fass and Y. Wilks. Preference semantics, ill-formedness, and metaphor. *American Journal of Computational Linguistics*, 9(3-4):178–187, 1983. URL https://www.aclweb.org/anthology/J83-3004.

M. Honnibal and I. Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

L. Huang, C. Sun, X. Qiu, and X. Huang. Glossbert: Bert for word sense disambiguation with gloss knowledge, 2019.

I. Iacobacci, M. T. Pilehvar, and R. Navigli. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 897–907, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1085. URL https://www.aclweb.org/anthology/P16-1085.

A. Kilgarriff. What is word sense disambiguation good for?, 1997.

Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations, 2019.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

M. Majlis. Wikipediaapi. hhttps://github.com/martin-majlis/Wikipedia-API, 2017.

T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality, 2013.

G. A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, Nov. 1995. ISSN 0001-0782. doi: 10.1145/219717.219748. URL https://doi.org/10.1145/219717.219748.

R. Navigli and S. P. Ponzetto. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217 – 250, 2012. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2012.07.001. URL http://www.sciencedirect.com/science/article/pii/S0004370212000793.

R. Navigli, K. C. Litkowski, and O. Hargraves. SemEval-2007 task 07: Coarse-grained English all-words task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 30–35, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/S07-1006.

J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations, 2018.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2018. URL https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

J. Ramos. Using tf-idf to determine word relevance in document queries, 2013.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding, 2018.

Y. Wang, M. Wang, and H. Fujita. Word sense disambiguation: A comprehensive knowledge exploitation framework. *Knowledge-Based Systems*, 190:105030, 2020. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2019.105030. URL http://www.sciencedirect.com/science/article/pii/S0950705119304344.

G. Wiedemann, S. Remus, A. Chawla, and C. Biemann. Does bert make any sense? interpretable word sense disambiguation with contextualized embeddings, 2019.

Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2019.

D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA, June 1995. Association for Computational Linguistics. doi: 10.3115/981658.981684. URL https://www.aclweb.org/anthology/P95-1026.

D. Yuan, J. Richardson, R. Doherty, C. Evans, and E. Altendorf. Semi-supervised word sense disambiguation with neural models, 2016.

G. K. Zipf. Human behavior and the principle of least effort: An introduction to human ecology. by george kingsley zipf. cambridge, mass.: Addison-wesley press, inc., 1949. 573 pp. $6.50. 1950.

Apéndices

A

Dataset de frases en inglés

A continuación se muestra el dataset con el que se han ejecutado los experimentos; está formado por 75 frases cortas en inglés en las que existe una palabra polisémica (keyword) dentro de la frase.

Keyword	Phrase
arm	He has an injure in his arm
arm	ARM is inside this computer
arm	This shot came from that arm
apple	Apple is fantastic for your health
apple	Apple was the company behind The Beatles
queen	Freddie was the lead singer of Queen
queen	The Queen lives in Buckingham palace
congress	Donald Trump will testify in congress
tesla	Tesla created a lot of inventions that worked with electricity
tesla	Tesla is developping new car batteries
solution	The problem asked to find the solution of the equation
solution	Boil the solution with salt
key	I can't open the door cause i've lost my key
key	Cryptography is all about finding the key
key	You need to hash the key to map the value in the table
mouse	Computer does not include mouse
mouse	The mouse eat the cheese
table	You can find information stored on that table
table	I left the book on the table
crane	I saw that crane at the construction site
crane	That crane is an endangered species
wood	That table is made of wood
wood	Once you enter the green you will need to use a wood
green	The hole is inside the green
green	Green is the ecologist color

letter	You can find my address in that letter
letter	You can choose any letter in the English alphabet
orange	I usually drink orange juice with my breakfast
orange	Orange is the new black
orange	Orange is a very peaceful city
orange	Orange offers good mobiles
dog	The dog ate the cat food
dog	He was eating a hot dog
goal	His determination will help him to achieve his goal
goal	The player scored a goal
rock	She is so into Queen, she loves rock
rock	I saw a big rock in the geology lab
car	We had to change a tire of my car
cars	We went to the movies and saw Cars
cat	I've got a dog and a cat
cat	Computers are advancing in CAT on every language
bass	I can hear the bass sound in the band
bass	He loves grilled bass with potatoes
pen	I need a pen to continue my writting
pen	All the sheeps where in the pen
pen	He was in the pen for murder
horse	He knew how to ride a horse
horse	She jumped over the horse at the gym
knight	The castle had a knight guarding it
knight	He moved his knight next to the pawn
yesterday	I didn't have the time yesterday, I'll try it today
yesterday	Yesterday is an amazing song
break	Nadal is playing a break point
break	I'm having a break from work tomorrow
queen	He moved the queen and it was check
head	He is the head of the government
head	She has an injure at the back of her head
field	Many animals can be found in this vast field
field	The quotient field of this algebraic extension
field	Maxwell proved that there is always a magnetic field induced by an electrical current
field	I am particularly interested in the study of that field
bar	After the LSAT he needs to pass the Bar
bar	The policeman entered the bar and asked for a drink

flash	Flash and Superman are both superheores
flash	That flash was so bright it blinded me
scale	The scale on this map is confussing me
scale	I can play a major scale with my piano
string	String theory describe electrons as a lot of tiny moving particles
string	The violin is a string instrument
hammer	I need a hammer for this nails
hammer	Ear damage includes hammer fractures
ray	A ray can only cross another one in one point
ray	I saw a ray in the ocean while I was swimming
mouth	The dentist told him to open his mouth
mouth	The water is fresh at the mouth, on that mountain.

B

Librería de parseo de *Phrasal Verbs*

En este apéndice se muestra la librería de reconocimiento de phrasal verbs en inglés.

Código B.1: Librería de Phrasal Verbs I

```
import spacy
import en_core_web_lg
from nltk.stem.porter import *
from more_itertools import locate
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.porter import PorterStemmer
import string
import numpy as np
from nltk.stem import WordNetLemmatizer
stemmer = SnowballStemmer(language='english')
lemmatizer = WordNetLemmatizer()
wiki = wikipediaapi.Wikipedia('en')
nlp = en_core_web_lg.load()
stopwords = "stopwords.txt"
# Diccionario de reglas para phrasal verbs
#
rules_dict=[
                (["VERB","ADV","ADP"],["VERB","ADV","ADP"]),
                (["VERB","NOUN-CHUNK","ADP"],["VERB","#","ADP"]),
                (["VERB","ADP"],["VERB","ADP"]),
                (["VERB","ADV"],["VERB","ADP"])
]
def group_matches(chunk_list,rules):
    """

    Dada una lista de tokens & noun_chunks de SpaCy y un conjunto de reglas
    reestructura la lista de noun_chunks en grupos segun las reglas

    Arguments:
        chunk_list {list} --[description]
        rules {list[tuples]} --[description]

    Returns:
        [list]--lista de tokens asociada segun reglas
    """
```

Código B.2: Librería de Phrasal Verbs II

```
36      work_list = []
37      for chunk in chunk_list:
38          try:
39              work_list.append(chunk.pos_)
40          except AttributeError:
41              work_list.append("NOUN-CHUNK")
42      chunk_list = get_words(chunk_list)
43      for rule in rules:
44          matching = match(work_list,rule[0])
45          if matching != None:
46              for index in matching:
47                  chunk_list,work_list = compress_in_one(chunk_list,work_list,index,rule)
48      return chunk_list
49
50  def get_words(l):
51      """
52      Convierte los tokens que no son noun_chunks de una lista
53
54      Arguments:
55          l {list} --lista de tokens (Spacy.Tokens o Spacy.noun_chunks)
56
57      Returns:
58          [list] --lista de Spacy.noun_chunks y str done antes habia Spacy.Tokens
59      """
60      s = l.copy()
61      for i in range(len(s)):
62          try:
63              s[i] = s[i].text
64          except AttributeError:
65              continue
66      return s
67
68
69  def compress_in_one(l,l_aux,index,rule_tuple):
70      """
71
72      Sustituye un elemento de una lista por su match
73
74      Arguments:
75          l {list} --lista de trabajo
76          l_aux {list} --lista con las posiciones sintacticas de los elementos de l
77          index {int} --indice donde va a empezar la sustitucion
78          rule_tuple {tuple} --regla de sustitucion
79
80      Returns:
81          [tuple] --new_list,new_work_list
82      """
83      rule,subs = rule_tuple
84      sub,others,others_pos = substitude(l[index:index+len(rule)],l_aux[index:index+len(rule)],subs)
85      l[index] = sub
86      l_aux[index] = "MATCH"
87      l = l[0:index+1]+others+l[index+len(rule):]
```

Código B.3: Librería de Phrasal Verbs III

```
88      l_aux = l_aux[0:index+1]+others_pos+l_aux[index+len(rule):]
89      return l,l_aux
90
91  def substitude(l,l_aux,subs):
92      """
93
94      agrupa los elementos que va a sustituir y separa las listas originales
95
96      Arguments:
97          l {list} --lista de elementos original
98          l_aux {list} --lista de posiciones sintacticas
99          subs {list} --forma de la sustitucion (lista gramatical)
100     """
101     in_subs = []
102     out_subs = []
103     out_pos = []
104     for i,x in enumerate(l):
105         if l_aux[i] == subs[i]:
106             in_subs.append(x)
107         else:
108             out_subs.append(x)
109             out_pos.append(l_aux[i])
110     return "␣".join(in_subs),out_subs,out_pos
111
112 def match(l,pattern):
113     """
114
115     devuelve una lista con los indices de matching de un patron en una lista
116
117     Arguments:
118         l {list} --lista de elementos
119         pattern {list} --lista con el patron deseado
120
121     Returns:
122         [list] --lista con todos los indices en los que se ha encontrado el patron
123     """
124     length = len(pattern)
125     poss = [l[i:i+length] for i in range(0,len(l)-length+1)]
126     if pattern in poss:
127         return list(locate(poss,lambda a: a == pattern))
128     else:
129         return None
130
131 def lemmatize(sentence):
```

Código B.4: Librería de Phrasal Verbs IV

```
138         """
139
140         deveulve los sintagmas lemmatizados de una frase
141
142         Arguments:
143             sentence {str} --a sentence
144
145         Returns:
146             [list] --lista con los tokens de SpaCy lemmatizados
147         """
148         l = []
149         for word in sentence:
150             tokens = nlp(word)
151             l.append("␣".join([token.lemma_ if token.lemma_ != "-PRON-" else token.text for token in
                    tokens ]))
152         return l
153
154     def unify_noun_chunks(sentence):
155         """
156
157         Unifies the noun chunks of a sentence and the SpaCy tokens
158
159         Arguments:
160             sentence {str} --some sentence
161
162         Returns:
163             [list] --unified list
164         """
165         analysis = nlp(sentence)
166         ret,aux = [],[]
167         chunks = [chunk for chunk in analysis.noun_chunks]
168         count = 0
169         start,end = False,False
170         for token in analysis:
171             if token not in chunks[count]:
172                 ret.append(token)
173                 if start:
174                     start= False
175                     count = (count + 1) % len(chunks)
176             elif chunks[count].text not in aux:
177                 ret.append(chunks[count])
178                 aux.append(chunks[count].text)
179                 if not start:
180                     start = True
181             else:
182                 continue
183
184         return ret
```

Código B.5: Librería de Phrasal Verbs V

```
185
186 def nodes(sentence):
187     """
188
189     Returns the concepts that would act as nodes in an NLP algorithm in a list
190
191     Arguments:
192         sentence {[list} --list with nodes
193     """
194     l = unify_noun_chunks(sentence)
195     l = group_matches(l,rules_dict)
196     return [x for x in lemmatize(l) if len(x.split("␣")) != 1 ]
```

C

Diagrama de funcionamiento general

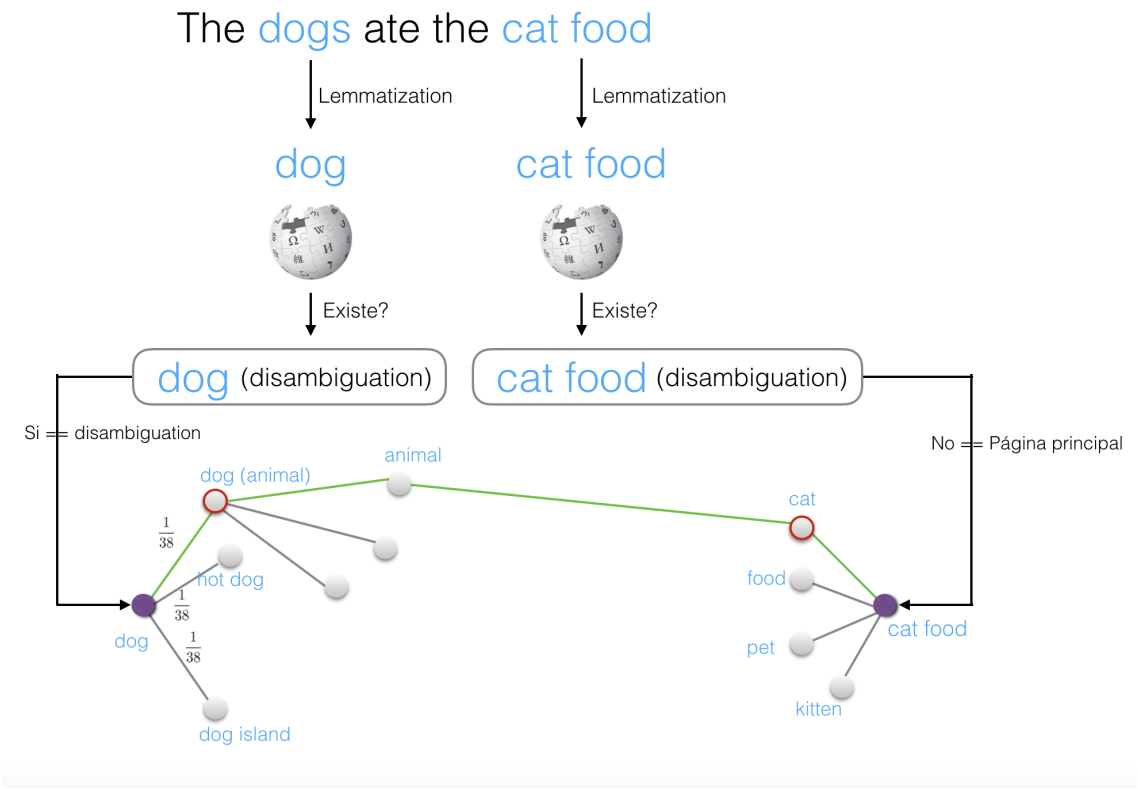


Figura C.1: Diagrama general

1.– Se toman los sustantivos (de momento) de la frase.

2.– De cada uno se mira si está compuesto por una o más palabras (noun_chunk)

3.– Para cada sustantivo miramos si existe la pagina de desambiguación

4.– Si la página existe se le asigna un nodo con la página de desambiguación (i.e. es un concepto ambiguo). Si no, no es ambiguo y se le asigna un nodo con la página principal asociada

5.– Mientras exista un nodo raíz de un árbol (en la foto en morado) que no pertenezca a la misma componente conexa que otro, expandir el par de nodos que diga la heurísitica


UAM
UNIVERSIDAD AUTONOMA
DE MADRID